Connect the Magic

An Introduction to the WIZnet W5500

Are you ready to join the Internet of Things (IoT) revolution? You can start by building an innovative 'Net-enabled design to enter in the WIZnet Connect the Magic Challenge. Before you begin, read through this comprehensive introduction to WIZnet's W5500 "smart" Ethernet chip and the WIZ550io integrated module.

By Tom Cantrell (US)

WIZnet Connect the Magic Challenge: March 3, 2014 – August 3, 2014

Visit circuitcellar.com/wiznet2014

- *Request Free WIZ550io Modules*
- *How to Register & Enter*
- *Rules & Regulations*
- *Prize Info*
- *And more*

The microelectronics revolution is entering a new phase of mass market acceptance and application. I'm talking about the proliferation of popular low-cost single-board computers (SBCs) that let anyone craft their own embedded application. Like magic, these platforms turn mere "users" into "makers."

These magic microcontrollers all share certain characteristics. The hardware itself is low cost with a range of products and add-ons; the tools (i.e., compiler and IDE) are free and easy to use; and, most importantly, there's a self-sustaining "community" leveraging shared knowledge for the benefit of all.

I've had the pleasure of working with popular platforms such as Arduino, Texas Instruments's LaunchPad, ARM mbed, Parallax, and others (see **Photo 1**). All of them make it quick and easy, and dare I say fun, to whip out surprisingly sophisticated applications. And all have vibrant user groups with plenty of useful resources (tools, examples, advice) to share.

Now it's time to connect all these gadgets to the Internet of Things (IoT). Enter WIZnet and its latest and greatest "smart" Ethernet chip. The W5500 (see **Figure 1**) starts with a standard 10/100 Ethernet interface (i.e., MAC and PHY) but then goes further with large RAM buffers (16-KB transmit and 16-KB receive) and hardware TCP/IP protocol processing.

I discovered WIZnet's first chip, the W3100, way back in 2001 (see my article "I-Way the Hard Way," *Circuit Cellar* 135, 2001). Of course by now, as with all things silicon, the new W5500 is better, faster, and lower cost. But the concept is still exactly the same: "Internet enable" applications by handling the network chores in hardware so the application microcontroller doesn't have to do it in software.

The large RAM buffers help decouple the microcontroller from network activity. In a recent project (see my article, "Weatherize Your Embedded App," *Circuit Cellar* 273, 2013), I used the RAM to receive an entire 10-KB+ webpage, completely eliminating the need for the microcontroller to juggle data at network speed. And any of the 32-KB on-chip RAM that isn't needed for network buffering is free for general-purpose use, a big plus for typically RAM-constrained microcontrollers.

The other major WIZnet hardware assist is TCP/IP processing using IP addresses, sockets, and familiar commands including OPEN, CONNECT, SEND, RECEIVE, DISCONNECT. The high-level interface to the network frees up microcontroller cycles and code space that would otherwise be needed for a software TCP/IP stack.

DISAPPEARING ACT

The WIZ550io (see **Photo 2**) is an integrated module that includes everything you need to get online.

Connecting the WIZ550io to your favorite microcontroller is easy. There's just a SPI

(MISO, MOSI, SCLK, nSCS), three status/control lines (nRESET, RDY, nINT), and power and ground. The WIZ550io power supply is 3.3 V, but the module inputs are all 5-V tolerant.

Ideally your processor has a hardware SPI port that can take advantage of the WIZnet module's high-speed (up to 80 MHz) SPI. But if not, bit banging is fine since the W5500 RAM buffers will take up the slack for a slow microcontroller connection.

As for the three control lines, you can connect them or not, depending on the particulars of your application.

nRESET does a hardware reset of the module, but the automatic power on reset will typically suffice. Alternatively, you can do a little housekeeping (save the current network parameters) and issue a software reset.

After hardware reset (i.e., power on or nRESET), the RDY output will assert after a delay (50 ms) for internal module initialization. Instead of dedicating a pin to monitor RDY, it's easy to just insert a software delay when the application starts.

The nINT pin is there if you want to implement an interrupt-driven interface. Software can define which particular event(s) (e.g., data transfer, socket disconnect, link loss, etc.) trigger an interrupt request. But with the W5500 handling most network activity, there's no need to interrupt the microcontroller in normal operation. The network can be dealt with in the background, leaving interrupts free for real-time tasks that truly need them.

How many times have you gotten near the end of a project and discovered you really need one more I/O line? The W5500 offers an optional fixed transfer length SPI mode that works with the chip select (nSCS) input grounded. However, fixed mode only supports short transfers (1, 2, or 4 bytes), not the arbitrarily large block transfers possible using nSCS, so only consider it as a last resort if you're really desperate to free up a pin.

HIDDEN WIRES

Blasting data fast and far is Ethernet's claim to fame, but that requires a lot of power (e.g., 100+ mA) just to maintain the link (i.e., PHY enabled). Fortunately, the W5500 has a standby mode that drops the link (i.e., disables the PHY), reducing the W5500 power consumption by a factor of 10 as well as that of the un-linked partner.

Having an AC outlet nearby gives you the option of piggybacking your Ethernet data onto the power lines. That's exactly what I do with my own garage door monitoring "Thing" (see **Photo 3**). Shop around and you'll find

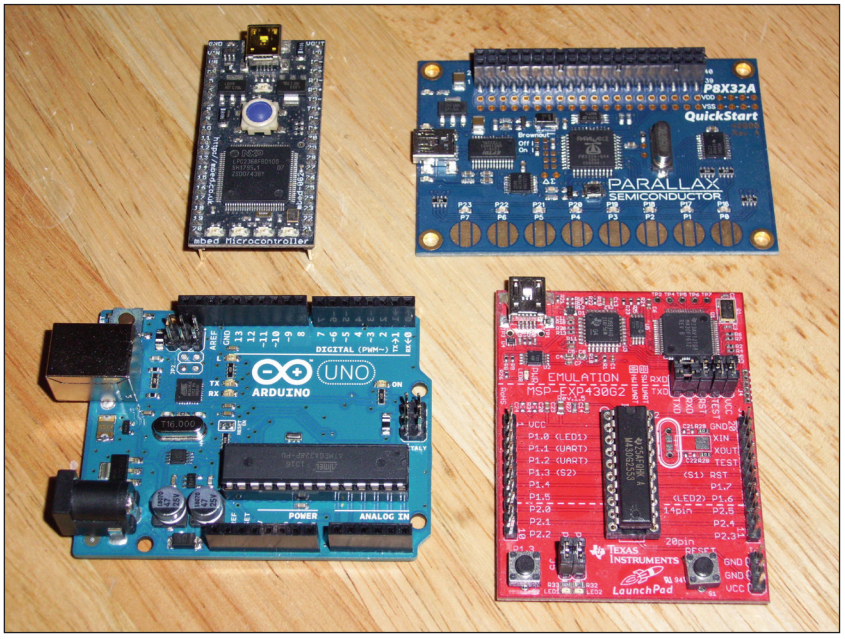

PHOTO 1
Magic tricks are easy when you've got the right cards up your sleeve.

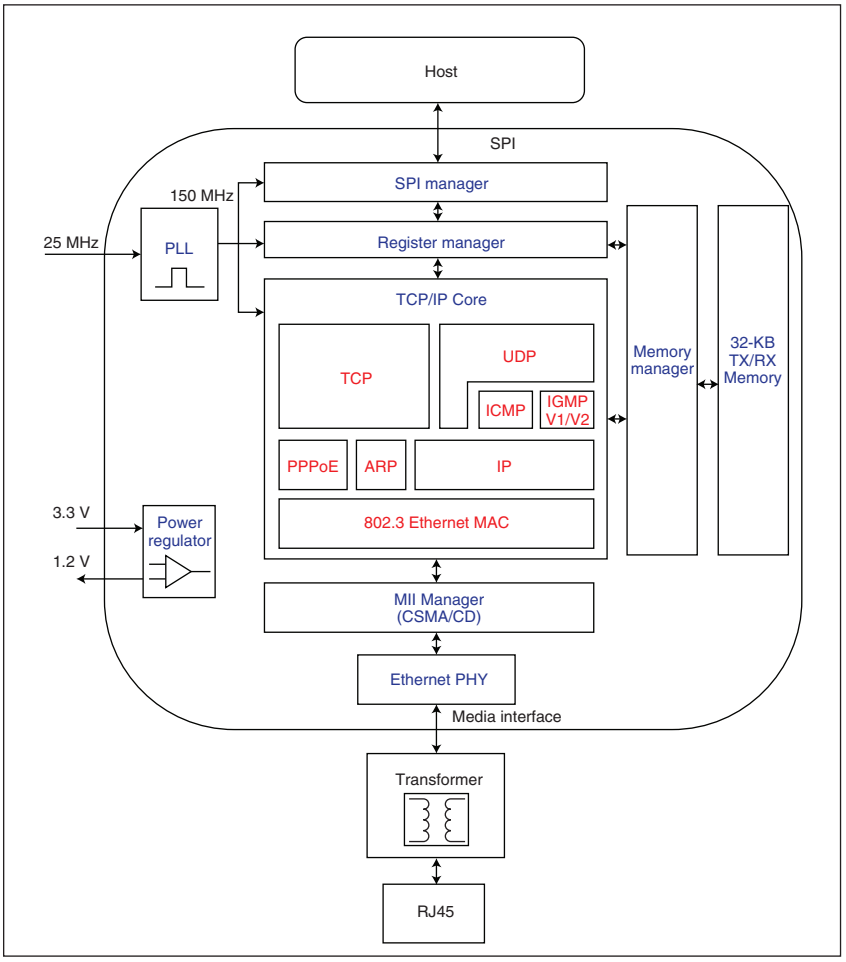

FIGURE 1
The WIZnet W5500 is an Ethernet chip with a difference—large RAM buffers and hardware TCP/IP processing that make it easy for any microcontroller to go online.

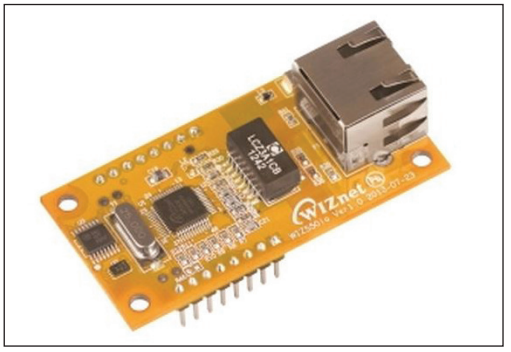

PHOTO 2
The WIZ550io module has everything you need (e.g., transformer, RJ45, and MAC address) to plug and play.

PHOTO 3
My garage door monitoring "Thing" uses a powerline adapter to connect with the household LAN.

there are powerline communication "starter kits" that come with a pair of adapters (such as my Rosewill RPLC-201KIT) in the $20 to $40 price range.

If you're recalling the shaky early days of power line communication, be advised the latest generation of gear works a lot better. There can still be situations where particular AC outlets are hard to reach, but usually you can find a nearby one with a decent connection.

PHOTO 4
All you need are some cheap "cheater" cables (splitter and injector) and a variable power supply to homebrew your own "Passive PoE" solution.

Since these new adapters are designed to handle streaming AV, I've found that even marginal connections (as indicated by the adapters signal quality LED) work fine with my low-bandwidth apps. Presumably the powerline adapter and/or W5500 are working their own magic handling any issues (e.g., automatic retry of corrupted/lost packets,) which is fine by me.

If there isn't AC nearby and you're faced with stringing wire, Power over Ethernet (PoE) is the way to go. It's perfect for things such as security webcams and VoIP phones, and now the popularity of those applications has fueled the market with new suppliers and lower prices (less than $100 routers and less than $10 modules) for IEEE standard 802.3af PoE gear.

An even simpler roll-your-own option is "Passive PoE," which uses the four spare wires in a standard Ethernet cable for power transmission (see **Photo 4**). Note that this hack doesn't work with Gigabit Ethernet (which uses all eight wires) or "Active PoE" (i.e., IEEE standard 802.3af) gear, just plain vanilla 10/100.

You may have to consider voltage drop, especially for long cable runs (iup to

100 m) and/or high-current loads. Use a "PoE Calculator," such as the one available from Stephen Foskett's blog (see Resources), to estimate the voltage drop depending on your cable length and current requirements

Or just wire everything up and dial it in using a variable power supply while measuring the actual voltage at the far end. Make sure your device is attached and fully active (i.e., pulling maximum expected current, not sleeping, etc.) to get an accurate voltage reading.

MIND READER

If your heart is set on wireless, you can always add a low-cost Wi-Fi adapter. The TP-Link NanoRouter I've been using (see **Photo 5**) is quite versatile with five different configuration options: Router, Access Point, Bridge, Repeater, and Client. Your attached Ethernet device can either join an existing wireless network or you can create an additional network with its own SSID.

Thanks to the march of technology, the dual Ethernet + Wi-Fi approach is getting more affordable. The WIZnet online store has the WIZ550io for $18 (the W5500 chip costs just $2.87) and you can find TP links online for $20 or so. By comparison, if you Google "embedded Wi-Fi module" or "Wi-Fi shield," you'll see a variety of different products ranging from around $20 for a bare Wi-Fi module (no PCB) to $80 for an official Arduino WiFi Shield.

Indeed, the large variety of Wi-Fi solutions does have a downside. There are more than a few embedded Wi-Fi chipsets in popular use (e.g., Broadcom, GainSpan, Microchip Technology, Texas Instruments, no doubt more on the way), each with their own capabilities and unique commands. The prospect of trying to write and support drivers for all the Wi-Fi chipsets across all the popular platforms gives me a headache. With the dual approach, it's easy to move Ethernet apps (any app, any platform) to Wi-Fi. Just plug into the TP link and you're done without changing a line of code—that's the kind of magic trick I like!

SERVER UP MY SLEEVE

Everyone wants to be master of the IoT universe from their browser screen. With assistance from WIZnet, even tiny microcontrollers can run a simple web server. Check out the webpage served up by my garage door "Thing" (see **Photo 6**).

But you can see the problem; or rather you can't see it. Where are the high-resolution graphics that a pixel-jaded populace demands? No way to cram more than a bit of HTML/JavaScript/JPG eye candy

PHOTO 5
Low-cost mobile hot spots such as this TP-Link NanoRouter make it easy to convert Ethernet to Wi-Fi without any software changes.

PHOTO 6
My garage door "Thing" doing its thing.

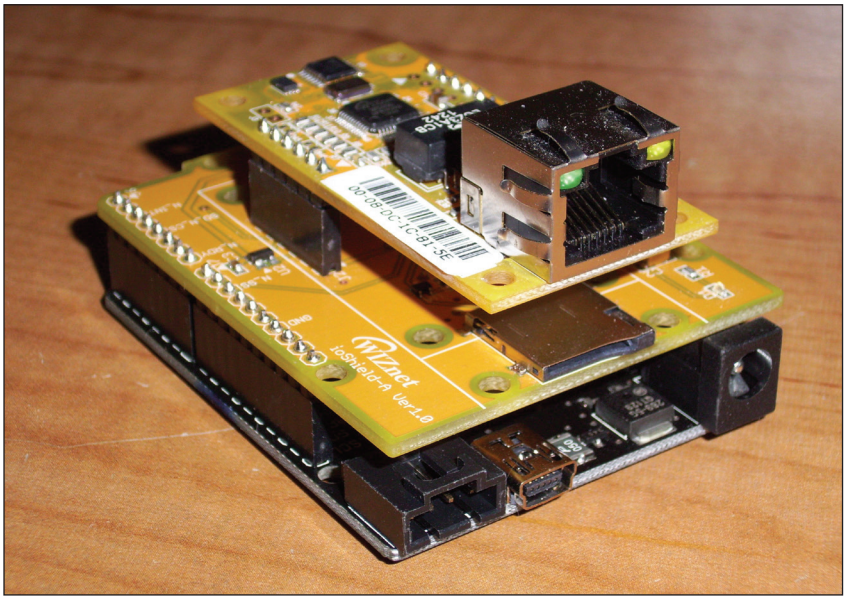

PHOTO 7
If you want a fancy server with lots of eye candy, a microSD card is the way to go. The WIZnet ioShields include the card socket and are available for various platforms. The Arduino version is shown here.

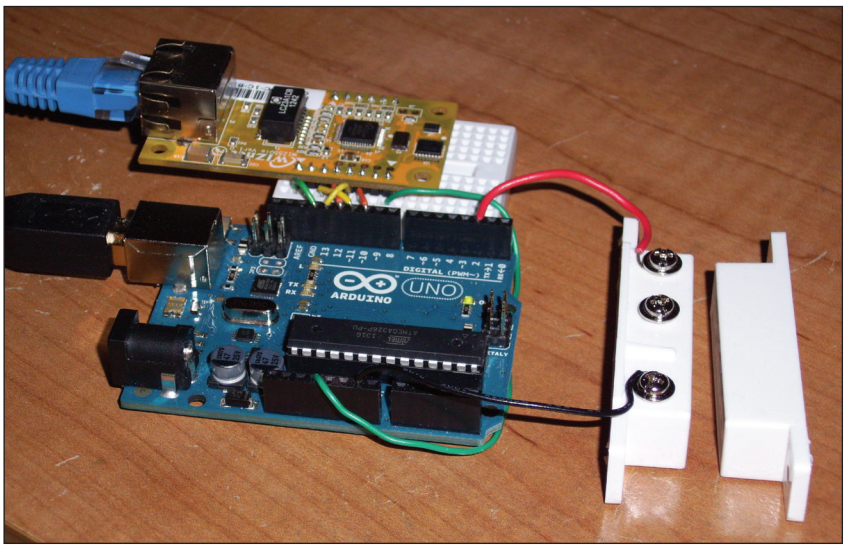

PHOTO 8
A prototype of the client version of my garage "Thing" is shown.

ABOUT THE AUTHOR

Tom Cantrell (microfuture@att.net) has been working on chip, board, and systems design and marketing for several years.

into the microcontroller itself, so you'll have to conjure up some external memory if you want to spice things up.

The most popular add-on is a MicroSD card, which, like the W5500, uses a SPI bus so you just need one extra pin for chip select. Using a standard file system driver (e.g., FAT), you can actually do much of the development and testing of your "website" on a PC. Then when you're finished, just plug the SD card into your IoT gadget.

For prototyping, check out the WIZnet ioShield (see **Photo 7**), which is a baseboard for the WIZ550io that includes an SD card socket. There are ioShields for different platforms (e.g., Arduino, LaunchPad, mbed, etc.), and with 0.1" headers they are breadboard friendly.

PRESTO, CHANGE-O

Since your browser is a "client," it makes sense that every IoT gadget should be a "server," right? Not necessarily. Being a server typically implies being open for service 24/7, but many IoT applications (e.g., my garage "Thing") are characterized by intermittent low duty-cycle activity. And while running on a LAN works fine, there may be issues reaching an in-house server from the WAN due to firewalls, ISP restrictions, changing IP addresses, and the like. Besides, do you really want to let the outside world anywhere near your LAN? Maybe it makes more sense for IoT gadgets to be clients.

But how do you get two clients (i.e., IoT device and browser) to talk to each other? The answer is you stick a server in between them courtesy of a "Device Cloud" provider such as Xively (formerly Pachube), Exosite, Devicehub.net, ThingSpeak, Nimbits, XOBXOB, the list goes on and on. (Postscapes provides an extensive list, see Resources.)

As much as the WIZnet chip offloads network chores, these services handle data storage and visualization to make life easy for the IoT application. Just send your raw data and the service will archive it, present it to a browser as a graphical chart, and send an e-mail, text, or Twitter alert if you like. Better yet, you don't need any specialized web programming tools or know-how to get something useful working quickly.

The technical capabilities and look and feel of each device cloud service differ, as do their business models, everything from for-profit and pay-to-play to free and open source. But from 50,000' all work in a similar way. To send data to the cloud, the IoT client hits the cloud server with a request that carries the data (e.g., variable name and value) in the URL or the request body. To get data from the cloud, the IoT device sends a request specifying a variable and then retrieves the data from the server response.

I decided to get my head in the clouds by prototyping a client version of my garage door "Thing" (see **Photo 8**) using an Arduino + a WIZ550io connected to Exosite.

Note I'm powering the WIZ550io from the Arduino 3.3-V supply. That works for newer Arduinos (e.g., my UNO R3) that have a 150-mA, 3.3-V regulator. To work with earlier Arduinos or clones that only specify 50 mA, a real "shield" (e.g., the WIZnet ioShield) will include a 3.3-V regulator running off the 5-V supply.

circuitcellar.com/ccmaterials

RESOURCES

T. Cantrell, "I-Way the Hard Way," *Circuit Cellar* 135, 2001.

——, "Weatherize Your Embedded App," *Circuit Cellar* 273, 2013.

S. Foskett's blog, "Power Over Ethernet Calculator," http://blog.fosketts.net/toolbox/power-ethernet-calculator.

Postscapes, "IoT Data Broker and Cloud Service Providers," http://postscapes.com/companies/iot-cloud-services.

WIZnet, "Connect the Magic," http://j.mp/ConnectTheMagic.

SOURCES

Exosite Device Cloud
Exosite | www.exosite.com

RPLC-201KIT AV adapter starter kit
Rosewill, Inc. | www.rosewill.com

TL-WR702N 150 Mbps Wireless N NanoRouter
TP-Link Technologies, Co. | www.tp-link.us

W5500 Ethernet controller
WIZnet | wizwiki.net

The Arduino code (see **Photo 9**) just sits in a loop checking to see if the door state changes or it's time to send a heartbeat. Then it takes little more than a single function call (`exosite.writeRead`) to fire the door state off into the Exosite cloud.

Over on the Exosite website, after signing up for a free "Developer" account, it was a quick and easy mainly point-and-click exercise to configure my "Device," "Data," "Events," and "Alerts" (see **Photo 10**).

As a client, there's no need to keep the "Thing's" Ethernet link powered all the time. Data only needs to be sent when the garage door opens or closes, but I also recommend sending a periodic heartbeat just in case. My garage door monitor will only generate a minute or two of network activity (i.e., door state changes and hourly heartbeats) per day, so there's opportunity for significant energy savings compared to a 24/7 server.

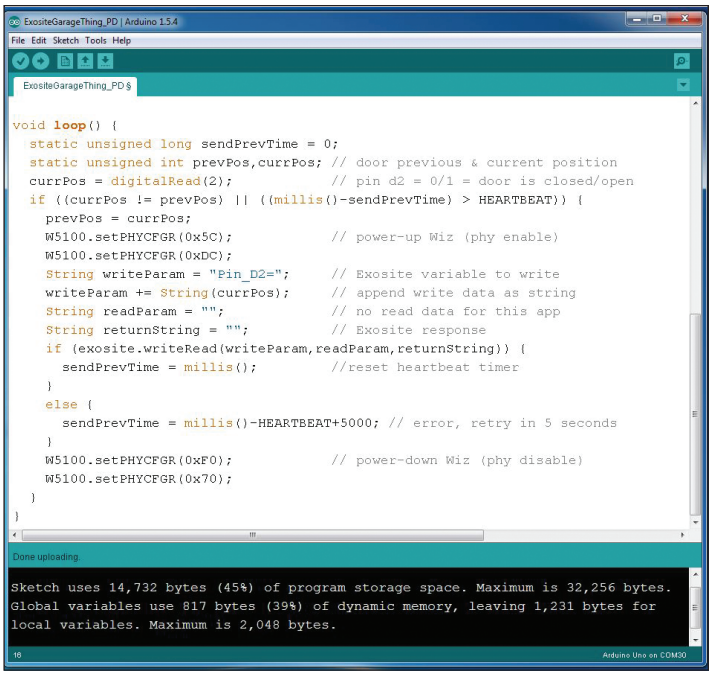

PHOTO 9
There's an Exosite library for Arduino that makes accessing the cloud as easy as can be.

LEARN SOME TRICKS

At WIZnet's "Connect the Magic" website you can find the props for your own magic show (see Resources). There's support for the WIZnet hardware (i.e., W5500, WIZ550io, and ioShields) as well as links to W5500 drivers and demos for third-party and open-source hardware including Arduino, LaunchPad, mbed, and Parallax (see **Photo 11**).

The W5500 also works with some interesting platforms I haven't used before. Cookie and chipKIT are Arduino form-factor SBCs that use ARM Cortex and Microchip Technology PIC32 microcontrollers, respectively. GR-KURUMI is a Japanese variation on the mbed theme (i.e., web-based tools) using a Renesas microcontroller. If you want to leverage existing big iron network software, there's even a BSD Sockets library based on the UC Berkeley open-source UNIX derivative.

The combination of magic micros with the W5500 and all the new device cloud services makes a compelling case for connection. Put 'em all in your hat, wave your magic wand, and amaze your friends with the cool "thing" you pull out.

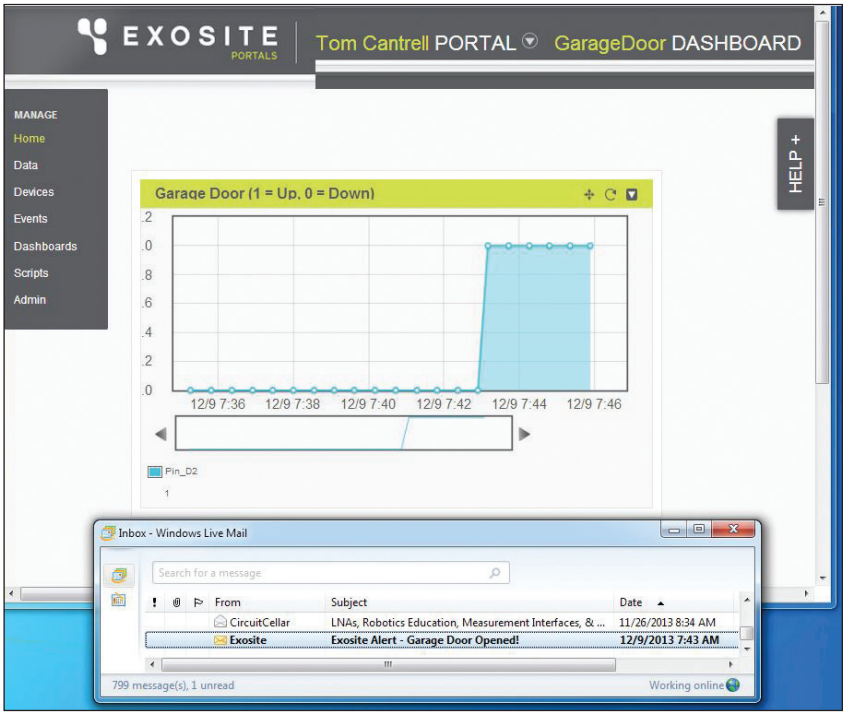

PHOTO 10
It only takes a few minutes to set up a simple Exosite dashboard including an e-mail alert. I can "see" my garage door without getting off the couch and now, via Exosite, from the farthest reaches of the web.

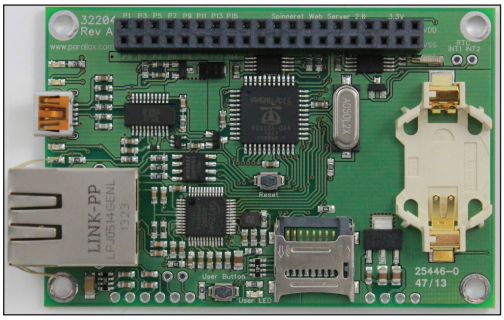

PHOTO 11
The Parallax Spinneret Web Server 2.0 is one of the first platforms available featuring the W5500.